EXPRESS MAIL LABEL NO.: EV268060904US DATE OF DEPOSIT: NOVEMBER 13, 2003

I hereby certify that this paper and fee are being deposited with the United States Postal Service Express Mail Post Office to Addressee service under 37 CFR § 1.10 on the date indicated below and is addressed to the Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

ROBERTA SHERMAN	[illegible]
NAME OF PERSON MAILING PAPER AND FEE	SIGNATURE OF PERSON MAILING PAPER AND FEE

Inventor(s): Scott Carrier

LIGHTWEIGHT FORM PATTERN VALIDATION

BACKGROUND OF THE INVENTION

Statement of the Technical Field

[0001] The present invention relates to the distribution of form-based input screens, and more particularly to the client-side validation of pattern conformity for form-based input provided through a distributed form.

Description of the Related Art

[0002] Form based input is the enabling technology which permits the widespread distribution of applications across generic client platforms such as the conventional content browser. In the prototypical distributed application, a markup language defined interface can form the principal conduit through which end users can interact with backend application logic. Often configured in the form of a Web page, the interface can be provided to the content browser by a content server and can take the form either of a pre-defined static page, or a dynamically generated page. Form input fields can be positioned within the interface through which user input can be accepted and posted to the backend application logic for further processing.

[0003] As in the case of any application, regardless of its mode of distribution, oftentimes, validation will be required for each input field in a form defined within an interface in which the end user can provide ad hoc input. Examples of input fields which might require validation can include free-form text input fields in which the end user is free to provide any time of input able to provided within the field. As the application logic may expect input of a particular format or pattern, however, validation will be required. In this regard, pattern validation refers to the inspection of user input to ensure that the input conforms to a particular pattern. Examples include date formats, time formats, credit card number formats, address formats, telephone number formats and the like.

[0004] Within the context of hypertext markup language (HTML) documents, it is well known to incorporate pattern validation within the interface. Specifically, to ensure that form based input comports with a particular pattern, some have incorporated simplistic programmatic scripts within the markup language document in which the pattern utilized to validate a form input field has been hard coded in the script. Upon the submission of form based input in the markup language document, the script can be executed to validate that the form input has a format which matches the pattern hard coded within the embedded script.

[0005] Despite the flexibility afforded by script based pattern validation within HTML defined forms, HTML defined forms mix data, logic and presentation in contravention of standard programming practices. In this regard, the well-known model-view-controller paradigm demands that each of data, logic presentation remain separable. In this way, though the presentation layer may change to suit the user interface requirements of an end user, the underlying logic layer need not also change. To accommodate the increasing complexity of transactions conducted through forms, the XForms specification has been proposed as a presentation independent way of handling interactive Web transactions. Significantly, the XForms specification separates data and logic from presentation in that XForms utilizes the extensible markup language (XML) for data transport and HTML for data display.

[0006] Advantageously, in the conventional XForms implementation, pattern validation can be implemented using heavy-weight Javascript libraries at the client side. Implementing client-side pattern validation can be important because in many circumstances it is desirable to validate form input without requiring a transaction between client and server. Yet, to accomplish the pattern validation of XForms, one must deploy a heavyweight Javascript validation library at the client. Where the client is a fixed, desktop unit enjoying substantial resources, hosting a heavyweight library will be of no consequence. The same cannot be said, however, of the resource limited client such as a mobile device.

[0007] Mobile device clients differ from conventional computing clients in that less computing resources are available for use by any client side logic and also in that a communicative link to backend application logic will not always remain available. Given both constraints associated with mobile device clients, it will be preferable to validate form based input prior to submitting the form based input to the backend application logic. Yet, to date pattern validation in the mobile device client strictly has been limited to server side validation primarily because mobile device clients lack the resources required to host a heavyweight validation library.

SUMMARY OF THE INVENTION

[0008] The present invention addresses the deficiencies of the art in respect to forms based processing of input and provides a novel and non-obvious method, system and apparatus for lightweight pattern based validation of forms based input in a distributed form. In accordance with the present invention, a lightweight pattern validation system can include a validation processor configured with a prototype interface for receiving both a field validation pattern and also form based input to be validated against the field validation pattern. The system further can include a validation script library packaging the validation process.

[0009] Notably, a library reference to the script library can be disposed within markup defining a form. In particular, the form can have at least one form based input field programmed for validation using the validation processor. A function call to the validation processor further can be disposed in the markup. The function call can have a configuration for passing a reference to a value in the form based input field for validation in the validation processor. Preferably, additional function calls to the validation processor can be disposed in the markup. As such, each additional one of the functional calls can have a configuration for passing a reference to a value in a corresponding form based input field for validation in the validation processor. In each case, however, a validation shell function can encapsulate the function call or calls.

[0010] A pattern validation method also is contemplated within the scope fo teh present invention. In a preferred aspect of the invention, a pattern validation method can include retrieving a value for a form based input field from a form defined in markup rendered in a content browser. The retrieved value along with a validation pattern for the form based input field can be passed to a validation process disposed within a lightweight validation library coupled to the rendered markup. Subsequently, the retrieved value can be validated in the content browser according to the validation pattern. Notably, each step of retrieving, passing and validating can be repeated for at least one additional value for at least one additional form based input field disposed in the markup rendered in the content browser. In this regard, the retrieving, passing, and validating steps can be performed in a validation shell function disposed in the markup rendered in the content browser.

[0011] Additional aspects of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The aspects of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims. It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The accompanying drawings, which are incorporated in and constitute part of the specification, illustrate embodiments of the invention and together with the description, serve to explain the principles of the invention. The embodiments illustrated herein are presently preferred, it being understood, however, that the invention is not limited to the precise arrangements and instrumentalities shown, wherein:

[0013] Figure 1 is pictorial illustration of a pattern validation system which has been configured in accordance with the present invention;

[0014] Figure 2 is a schematic diagram of a markup language document which has been configured for use in the system of Figure 1; and,

[0015] Figure 3 is a flow chart illustrating a process for lightweight, client side pattern validation.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0016] The present invention is a system, method and apparatus for the pattern validation of forms based input in a distributed form. In accordance with the present invention a value for a form based input field can be retrieved from a form defined in markup rendered in a content browser. The retrieved value along with a validation pattern for the form based input field can be passed to a validation process disposed within a lightweight validation library coupled to the rendered markup. Subsequently, the retrieved value can be validated in the content browser according to the validation pattern. Notably, the retrieval, passage and validation can be repeated for at least one additional value for at least one additional form based input field disposed in the markup rendered in the content browser.

[0017] Figure 1 is pictorial illustration of a pattern validation system which has been configured in accordance with the present invention. As shown in Figure 1, a form 160 can be defined within a markup language document disposed within a content server 110. The content server 110 can be communicatively coupled to one or more client devices 130 over the computer communications network 120. Though only a single client computing device 130 has been shown for simplicity of illustration, the skilled artisan will recognize that a multiplicity of computing clients can be coupled to the content server 110 which can range from fully functional, robust personal computers to limited function, mobile devices including cellular telephones and personal digital assistants.

[0018] A pattern validation processor 150 can be defined to validate user input against a defined pattern as is known in the art of XForm pattern validation. Specifically, the pattern validation processor 150 can expose a prototype interface through which a specific pattern and a value can be provided to the routine for processing. The routine defined within the pattern validation processor 150 can parse the value to ensure that the value has a format which matches the specific pattern. Where the value matches the specific pattern, the routine can return an indication as such. Conversely, where the value does not match the specific pattern, the routine similarly can return a failing indication. Importantly, the pattern validation processor 150 can be packaged within a script library 140. Preferably, the pattern validation processor 150 can be the only routine disposed within the script library 140 such that the script library 140 can be viewed as lightweight.

[0019] In operation, when one or more input fields defined within the form 160 have been rendered in a content browser operating within the client computing device 130, one can interact with the input fields, providing values for the input fields as required. When the end user chooses to submit the form for processing based upon the values provided for the input fields, each value can be passed to the validation processor 150 as can a corresponding pattern for which the value must comport. The validation processor 150 in turn can confirm or reject the value allowing for the form to properly handle the invalidation event. Notably, all of the foregoing process can be performed in the client computing device 130 and not in the server computing device 110. Moreover, as the validation processor 150 can be packaged within a lightweight validation script library 140, there will be no need to cache a complete copy of a conventional script library thus permitting the lightweight validation script library 140 to remain resident in the client computing device 130--even where the resources of the client computing device 130 are limited.

[0020] Figure 2 is a schematic diagram of a markup language document which has been configured for use in the system of Figure 1. In accordance with the present invention, a markup language document 200 can be configured with a form 230. The form 230 can define one or more input fields 250 in which user input can be provided. The input fields 250 can be defined such that validation will be required prior to submitting provided values in the input fields 250 to back end application logic. To avoid performing validation in the server, however, the validation can be performed inline in association with the markup language document 200.

[0021] In this regard, a reference 210 to a lightweight script validation library can be disposed within the markup language document 200. The lightweight script validation library itself can include a validation process which can assert compliance between a provided value and a provided pattern. In this way, the library can remain generic and flexible, yet small in size and resource requirements such that the library can be deployed in resource limited devices. Most importantly, the size of the library need not change regardless of the number of patterns provided against which values in the form 230 are to be validated.

[0022] In any case, a script 220 can be disposed within the markup language document 200 in which selected ones of the input fields can be coupled to individual function calls 240 to the validation process in the script validation library. Each individual one of the function calls 240 can include a value for the corresponding form based input field, and a pattern against which the value is to be evaluated. The script 220 itself can be activated upon the triggering of an event in which the values are to be submitted over the computer communications network for further processing.

[0023] In further illustration of the foregoing process, Figure 3 is a flow chart illustrating a method for lightweight, client side pattern validation. Beginning in block 310, input data can be accepted in a markup language defined form. In decision block 320, it can be determined whether the input data is to be submitted for processing. Once the data is determined to be submitted for processing, in block 330 the input data provided in selected ones of the input fields can be passed to a validation shell. The validation shell can be a single function defined within the markup language document in which multiple pattern validation operations for corresponding multiple input fields can be processed within a single function call.

[0024] In this regard, in block 340 the first input field can be selected for processing and in block 350, the pattern validation process can be invoked while passing into the pattern validation process two parameters: the value of the input field and a pattern against which the input field is to be tested. In block 360 the result of the pattern validation process can be received and reported within the validation shell. Notably, the reported result can be handled in countless ways ranging from a mere textual indication that the value for the particular input field failed to validate, to the invocation of a help-specific function. In any case, in decision block 370 if more input fields remain to be tested, in block 380 the next input field can be selected and subsequently processed in blocks 350 through 370. Finally, in block 390 the process can end.

[0025] The present invention can be realized in hardware, software, or a combination of hardware and software. An implementation of the method and system of the present invention can be realized in a centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system, or other apparatus adapted for carrying out the methods described herein, is suited to perform the functions described herein.

[0026] A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which, when loaded in a computer system is able to carry out these methods.

[0027] Computer program or application in the present context means any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form. Significantly, this invention can be embodied in other specific forms without departing from the spirit or essential attributes thereof, and accordingly, reference should be had to the following claims, rather than to the foregoing specification, as indicating the scope of the invention.